Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) ~~Method~~ A method for processing a complex request addressed to at least one SNMP agent (5) of a resource machine (2b) of a computer system (1) from a complex protocol manager (4) of an application machine (2a), the application (2a) and resource (2b) machines communicating through a network (3), each agent (5) managing attribute tables belonging to the resource machine (2b), the instances of the tables being referenced by identifiers comprising indexes, characterized in that it consists of:

- transforming a filter (F1) derived from a complex request from the manager (4) of the application machine (2a) into a simplified filter (F2) comprising only conditions on indexes, the filter (F2) corresponding to the following matching characteristics: the filter (F2) lets through all the SNMP requests whose responses could verify the filter (F1), but filters out all the SNMP requests whose responses cannot in any way verify the filter (F1);
- limiting the SNMP requests to those that comply with the filter (F2);
- transmitting said limited SNMP requests to the SNMP agent (5) of the resource machine (2b) through the network (3);
- applying the filter (F1) to the responses obtained to the SNMP requests;
- the method making it possible to process said complex request and to optimize the number of the SNMP requests transmitted through the network (3).

2. (currently amended) ~~Method~~ A method according to claim 1, characterized in that it consists of:

1) transforming the filter (F1) derived from the complex request into a simplified filter (F2);

2) determining the first potential instance that verifies the simplified filter (F2); the identifier just below the identifier of the potential instance determined is called the test identifier;

3) finding, using an SNMP request, the instance of the table having as its identifier the one that follows the test identifier. If no instance is found, the processing method is terminated. If an instance is found, the instance found is called the solution instance;

4) applying the complex filter (F1) to the solution instance; if the instance verifies the filter (F1), it is part of the response to the complex request processed;

5) determining the first potential instance whose identifier is higher than the identifier of the solution instance and that verifies the simplified filter (F2). If no instance is found, the processing method is terminated. If an instance is found, the identifier that is just below the identifier of the potential instance is called the test identifier and the method resumes with the third step.

3. (currently amended) ~~Method~~ A method according to claim 2, characterized in that it consists of obtaining, in the first step, the simplified filter with the form:

```
(OR
  (AND
        condition on index 1: $C1_{(1)}$
        condition on index 2: $C2_{(1)}$
        ...
        condition on index n: $Cn_{(1)}$
    )
    ...
    (AND
        condition on index 1: $C1_{(i)}$
        condition on index 2: $C2_{(i)}$
        ...
        condition on index n: $Cn_{(i)}$
    )
    ...
).
```

4. (currently amended) ~~Method~~ A method according to ~~either of claims 2 and 3~~ claim 2, characterized in that, if in the first step, after simplification, the filter is reduced to:

- only the TRUE condition, the table is scanned in its entirety;
- only the FALSE condition, no instance can work.

5. (currently amended) ~~Method~~ A method according to ~~any of claims 2 through 4~~ claim 2, characterized in that, in order to obtain the simplified filter F2, it consists of immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

6. (currently amended) ~~Method~~ A method according to ~~any of claims 1 through 5~~ claim 1, characterized in that, in order to obtain a simplified filter F2, it consists of:

- transforming the complex filter into a combination of conditions on the attributes joined by the logical operators HAND, OR and NOT;
- pushing the NOT operators to the leaves and deleting the double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index;
- gathering all the ORs at the route of the filter and simplifying again.

7. (currently amended) ~~Method~~ A method according to claim 6, characterized in that in order to delete the conditions X, it consists of replacing the conditions X and NOT X with the constant TRUE.

8. (currently amended) ~~Method~~ A method according to ~~either of claims 6 and 7~~ claim 6, characterized in that in order to simplify the operations, it consists of:

- replacing the AND and OR tests having only one operand with this operand;
- replacing the AND operations containing only TRUE operands with the constant TRUE and the OR operations containing only FALSE operands with the constant FALSE;

- removing the TRUE conditions from the other AND operations and the FALSE conditions from the other OR operations;
- replacing the OR operations containing at least one TRUE operation with the constant TRUE and the AND operations containing at least one FALSE operand with the constant FALSE;
- replacing the conditions that are always TRUE or FALSE with the constant TRUE or FALSE;

all of these simplification operations being applied as many times as it is possible to do so.

9. (currently amended) [[The]] A method according to ~~any of claims 2 through 8~~ claim 2 characterized in that, in the second step, it consists of concatenating the first a value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain the zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

10. (currently amended) ~~Method~~ A method according to claim 9, characterized in that, in the fifth step, it

consists of performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. … .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$;
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

11. (currently amended) ~~Method~~ A method according to ~~any of claims 2 through 10~~ claim 2, characterized in that in the second and fifth steps, it consists of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

12. (currently amended) ~~System~~ A system for processing a complex request addressed to at least one SNMP agent (5) of a resource machine (2b) of a computer system (1) from a complex protocol manager (4) of an application machine (2a), each agent (5) managing attribute tables belonging to the resource machine (2b), the instances of the tables being referenced by identifiers comprising indexes, the system comprising an integrating agent (6) that makes it possible to implement the processing method according to ~~any of claims 1 through 11~~ claim 1.

13. (previously presented) A method for processing a complex request addressed to at least one SNMP agent (5) of a resource machine (2b) of a computer

system (1) from a complex protocol manager (4) of an application machine (2a), wherein the complex request addressed to the agent (5) from the manager (4) comprises SNMP attributes managed by the agent (5) and capable of being represented by a filter (F1, F2) constituted by any number of conditions on any number of attributes, linked to one another by any number of Boolean operators (AND, OR, NOT, EX.OR, etc.) and the application (2a) and resource (2b) machines communicate through a network (3), each agent (5) managing attribute tables belonging to the resource machine (2b), the instances of the tables being referenced by identifiers comprising indexes, comprising:

- transforming a complex filter (F1) derived from the complex request addressed to agent (5) from the manager (4) of the application machine (2a) into a simplified filter (F2) comprising only conditions on indexes, and adapted to let through all the SNMP requests whose responses could verify the complex filter (F1), but filter out all the SNMP requests whose responses cannot in any way verify the complex filter (F1);
- limiting the SNMP requests to those that comply with the simplified filter (F2);
- transmitting said limited SNMP requests to the SNMP agent (5) of the resource machine (2b) through the network (3); and
- applying the complex filter (F1) to the responses obtained to the SNMP requests;
- to thereby process said complex request and to optimize the number of the SNMP requests transmitted through the network (3).

14. (previously presented) A method according to claim 13, wherein an identifier just below an identifier of the potential instance determined is a test identifier further comprising:

1) determining the first potential instance that verifies the simplified filter (F2);
2) using an SNMP request to find the instance of the table having as its identifier the one that follows the test identifier and if no instance is found, terminating the processing method, if an instance is found, naming the instance found a solution instance;
3) determining whether the solution instance is part of the response to the complex request processed by verifying the complex filter (F1) and upon verification of the complex filter (F1), applying the complex filter (F1) to the solution instance;
4) determining the first potential instance whose identifier is higher than the identifier of the solution instance and that verifies the simplified filter (F2) and terminating the processing method if no instance is found and if an instance is found, naming the identifier that is just below the identifier of the potential instance a test identifier and resuming the step of using the SNMP request to find the instance of the table having as its identifier the one that follows the test identifier..

15. (previously presented) A method according to claim 14 comprising in the step of transforming the complex filter (F1) into the simplified filter (F2) having the form:

```
(OR
  (AND
      condition on index 1: $C1_{(1)}$
      condition on index 2: $C2_{(1)}$
      ...
      condition on index n: $Cn_{(1)}$
  )
  ...
  (AND
      condition on index 1: $C1_{(i)}$
      condition on index 2: $C2_{(i)}$
      ...
      condition on index n: $Cn_{(i)}$
  )
  ...
).
```

16. (previously presented) A method according to claim 14, wherein in the first step, after simplification, the simplified filter (F2) is reduced to:

- only the TRUE condition, in which case the table is scanned in its entirety;
- only the FALSE condition, in which case no instance can work.

17. (previously presented) A method according to claim 15, wherein in the first step, after simplification, the simplified filter (F2) is reduced to:

- only the TRUE condition, in which case the table is scanned in its entirety;
- only the FALSE condition, in which case no instance can work.

18. (previously presented) A method according to claim 14, characterized in that, in order to obtain the simplified filter F2, immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

19. (previously presented) A method according to claim 15, characterized in that, in order to obtain the simplified filter F2, immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

20. (previously presented) A method according to claim 16, characterized in that, in order to obtain the simplified filter F2, immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

21. (previously presented) A method according to claim 17, characterized in that, in order to obtain the simplified filter F2, immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

22. (previously presented) A method according to claim 13 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;

- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

23. (previously presented) A method according to claim 14 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

24. (previously presented) A method according to claim 15 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

25. (previously presented) A method according to claim 16 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

26. (previously presented) A method according to claim 17 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

27. (previously presented) A method according to claim 18 characterized in that, in order to obtain a simplified filter F2,

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and

- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

28. (previously presented) A method according to claim 22, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

29. (previously presented) A method according to claim 23, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

30. (previously presented) A method according to claim 24, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

31. (previously presented) A method according to claim 25, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

32. (previously presented) A method according to claim 18, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;

- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

33. (previously presented) A method according to claim 23, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;

replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

34. (previously presented) A method according to claim 24, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions
- that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

35. (previously presented) A method according to claim 25, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

36. (previously presented) A method according to claim 26, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;

- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

37. (previously presented) A method according to claim 27, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

38. (previously presented) A method according to claim 28, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

39. (previously presented) A method according to claim 29, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

40. (previously presented) A method according to claim 30, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;

- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

41. (previously presented) A method according to claim 31, having AND and OR operations and characterized in that in order to simplify the operations, it consists of:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE and OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations and FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE and AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE and conditions that are always FALSE with a constant FALSE;

all of said latter simplification operations being repeated as many times as it is possible to do so.

42. (previously presented) A method according to claim 14 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \dots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

43. (previously presented) A method according to claim 15 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \dots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

44. (previously presented) A method according to claim 16 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \dots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

45. (previously presented) A method according to claim 18 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

46. (previously presented) A method according to claim 22 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

47. (previously presented) A method according to claim 28 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

48. (previously presented) A method according to claim 32 characterized in that the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

49. (previously presented) A method according to claim 42, characterized in that the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. … .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$;
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

50. (previously presented) A method according to claim 43, characterized in that the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index k < p, we take the value Ik with I1.I2. … .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$;
- for any index k > p, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

51. (previously presented) A method according to claim 44, characterized in that the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index k < p, we take the value Ik with I1.I2. … .In being the identifier of the solution instance;

- for the index p, we take the value $Jp_{(i)}$;
- for any index k > p, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

52. (previously presented) A method according to claim 45, characterized in that the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index k < p, we take the value Ik with I1.I2. … .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$;
- for any index k > p, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

53. (previously presented) A method according to claim 14 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the

solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

54. (previously presented) A method according to claim 15 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

55. (previously presented) A method according to claim 16 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

56. (previously presented) A method according to claim 18 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

57. (previously presented) A method according to claim 22 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

58. (previously presented) A method according to claim 28 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

59. (previously presented) A method according to claim 32 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

60. (previously presented) A method according to claim 42 characterized in that the steps of determining the first potential instance that verifies the simplified

filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

61. (previously presented) A method according to claim 49 characterized in that the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

62. (previously presented) A system for processing a complex request comprising at least one SNMP agent (5) of a resource machine (2b) of a computer system (1) to which the complex request is transmitted from a complex protocol manager (4) of an application machine (2a), each agent (5) managing attribute tables belonging to the resource machine (2b), instances of the tables being referenced by identifiers comprising indexes, the system comprising an integrating agent (6) for processing the complex request,

means for transforming a complex filter (F1) derived from the complex request addressed to agent (5) from the manager (4) of the application machine (2a) into a simplified filter (F2) comprising only conditions on indexes, the complex filter (F2) adapted to let through all SNMP requests whose responses could verify the simplified filter (F1), but filter out all SNMP requests whose responses cannot in any way verify the simplified filter (F1);

means for limiting SNMP requests to those that comply with the complex filter (F2);

means for transmitting said limited SNMP requests to the SNMP agent (5) of the resource machine (2b) through the network (3); and

means for applying the simplified filter (F1) to the responses obtained to the SNMP requests;

to thereby process said complex request and to optimize the number of the SNMP requests transmitted through the network (3).

63. (previously presented) The system for processing as set forth in claim 62 further comprising means for determining the first potential instance that verifies the simplified filter (F2) wherein the identifier first below the identifier of the potential instance determined is a test identifier.

64. (previously presented) The system for processing as set forth in claim 63 wherein, using an SNMP request, there is provided means to find the instance of the table having as its identifier the one that follows the test identifier and if no instance is found, terminating the processing method, if an instance is found, naming the instance found a solution instance; and means for determining whether the solution instance is part of the response to the complex request processed by verifying the complex filter (F1) and upon verification of the complex Sfilter (F1), applying the complex filter (F1) to the solution instance;

65. (previously presented) The system for processing as set forth in claim 64 further comprising means for transforming the complex filter (F1) into a simplified filter having the form

```
(OR
  (AND
        condition on index 1: $C1_{(1)}$
        condition on index 2: $C2_{(1)}$
        ...
        condition on index n: $Cn_{(1)}$
  )
  ...
  (AND
        condition on index 1: $C1_{(i)}$
        condition on index 2: $C2_{(i)}$
        ...
        condition on index n: $Cn_{(i)}$
  )
  ...
).
```